On uses of Extreme Value Theory fit for industrial-quality WCET analysis

Suzana Milutinovic[1,2], Enrico Mezzetti[1], Jaume Abella[1], Tullio Vardanega[3], Francisco J. Cazorla[1,4]
[1]Barcelona Supercomputing Center [2]Universitat Politècnica de Catalunya [3]University of Padova [4]IIIA-CSIC

***Abstract*—Over the last few years, considerable interest has arisen in measurement-based probabilistic timing analysis. The term MBPTA has been used to indistinctly refer to a variety of different applications of Extreme Value Theory (EVT) to the timing analysis problem. The successful application of MBPTA techniques to a score of case studies has not fully dispelled the concerns that industrial stakeholders had with the quality of the computed bounds, hence ultimately with their industrial viability. Placing focus on the MBPTA methods and techniques developed in the PROARTIS and PROXIMA projects, collectively referred to as proMBPTA, we discuss the main misconceptions and pitfalls that can prevent a sound application of EVT-based WCET analysis. Using a combination of arguments and support examples, we show that proMBPTA is a rigorous process, fully amenable to sound and sustainable industrial use.**

I. Introduction

The quest for novel WCET analysis methods capable of keeping pace with the trend towards more complex hardware has caused considerable interest to arise in statistical tools based on Extreme Value Theory (EVT) [1]. EVT considers the WCET as a rare event in the timing behaviour of a program and reasons about it in terms of probability of occurrence, yielding the notion of probabilistic WCET (pWCET).

Since EVT builds on observations, it is particularly attractive to real-time systems industry. In particular, it seams very naturally with measurement-based timing analysis (MBTA), which in spite of its theoretical limitations dominates industrial practice for WCET analysis. The Measurement-Based Probabilistic Timing Analysis (MBPTA) approach [2], [3] combines EVT's potential with the cost/benefit attractiveness of MBTA [4]. MBPTA much improves on standard MBTA by enabling the user to attach *quantitative confidence* to the pWCET bounds computed from timing measurements of the software program of interest, collected on the target platform.

Several works have consolidated various flavours of MBPTA [5]–[9], showcasing industrial-strength experiments run on processor simulators [10] or real hardware boards [11], [12]. However, before being deemed ready for transfer to industrial practice, MBPTA needs to be proven to: (i) deliver *trustworthy* results; (ii) be *adaptable* enough to embrace different kinds of software programs; and (iii) cause *lightweight* impact on an already effort-intensive verification and validation process. Arguably, the latter trait is a generally ascertained fact, with a score of industrial case studies [10]–[12] confirming that MBPTA requires an affordable number of measurement observations in addition to what normal practice already provides.

The *trustworthiness* of EVT has been challenged noting that the quality of its results critically reflects the quality of its inputs (for data, coverage, state control) [13], [14], with poor inputs yielding wholly unsound probability models. This should not surprise in fact, as the application of EVT to observations over user-controlled program runs instead of uncontrolled (e.g., natural) phenomena needs very careful attention and adaptation. In this paper we show that this misunderstanding originates from solely minding that the maxima of the input data (i.e. execution-time observations) pass EVT's all-famous pre-requisite tests of identical distribution and (sufficient) independence [6], [15]. However, it stands to reason that the measurement observations that one collects reflect the execution conditions that the program incurs (for input data, coverage of execution space, and state control) during analysis runs. In particular those conditions are intended to bear a sound relation with the worst execution conditions that can arise at operation. Only if that relation can be asserted, the analysis-time measurements can be said to be **representative**, thus useful for MBPTA. Ensuring representativeness is very complex indeed, for it requires controlling all sources of significant variation in the execution conditions explored in the analysis process.

The *adaptability* trait has been criticized too, on the argument that EVT-based analysis would not apply to all classes of software programs [13]. The critique showed that EVT was unable to produce good-fit distributions even for programs that exhibited sufficient execution-time variability, and was not applicable at all for those that had near-constant execution-time behaviour. This is where adaptation kicks in, which requires profound understanding of what differentiates the application of EVT to execution-time observations of program runs from its use in other domains, where the observed events and their sources have an altogether different nature.

In this paper we contend that these concerns are not inherent to EVT (or MBPTA) per se, but rather pertain to the way EVT is applied to the WCET domain, which may be seriously fallacious. We show that the MBPTA techniques developed in the PROARTIS and PROXIMA projects (collectively referred to as proMBPTA) solve the adaptation problem and can be used to produce trustworthy pWCET results.

II. Background on proMBPTA

MBTA (and MBPTA alike) aims at deriving high-quality estimates of the execution-time behaviour of selected programs during system *operation* by sampling measurement observations at various levels of system integration during *analysis*, i.e., testing. It is well known that the observable timing behaviour of a program reflects the specific execution conditions incurred in the measurement run, which are not guaranteed to stay the same from analysis to operation. In measurement-based approaches, this problem calls for some form of control by the user. Sufficient evidence for qualification or certification can be obtained, including for the highest criticality functions (e.g. DAL-A in avionics [16]), only as long as the user can prove ability to exercise exhaustive control over all the execution conditions considered of consequence. Looking at

978-1-5386-3166-9/17/$31.00

hardware only, the use of increasingly complex processors makes achieving the required level of control over the factors of influence (e.g., bus occupancy, data/code mapping in cache) much harder. This difficulty abates the confidence that can be placed in the computed estimates, and increases the effort intensiveness of the measurement collection, thereby reducing the cost/benefit ratio of MBTA dramatically.

The proMBPTA solutions lessen the swell in the user control burden considerably, while warranting soundness. To this end, they define the notion of MBPTA compliance [17], which reflects certain abilities of the observation process and specific features of the execution platform. The latter in particular require applying combinations of (hardware or software) randomization or upper-bounding to the execution-time behaviour of the hardware resources with the highest jitter. Applying upper-bounding at analysis ensures that what is observed upper-bounds the behaviour at operation. Applying time randomization ensures that what is observed at analysis is an equal representation of what can happen at operation. Time randomization can be realized directly in the hardware or obtained by software manipulations neutral to functional behaviour on top of COTS processors [8], [12], [17].

In its original fields of application, EVT is used to produce predictions on the extreme behaviour of the system from observations of it collected non-intrusively. To apply EVT to the execution-time domain, we must appreciate that there we observe the system in a non-neutral (hence biased) way as the user – intentionally or inadvertently – determines the execution conditions that occur during analysis, but may not do the same during operation. If different execution conditions may occur between analysis and operation, different (extreme) timing behaviours may emerge in the two situations: for EVT, they would describe two distinct systems, whose respective extreme behaviours may be far off one another. A simple-minded application of EVT to analysis-time observation data will thus produce pWCET estimates that *solely* upper-bound the execution time of the program *under the execution conditions captured in the analysis-time experiments*. Aspects related to the faulty hardware are not covered in this work, we refer interested reader to [18] for more informaton on this matter.

In order to ensure that the computed pWCET estimates hold at operation, proMBPTA creates a judicious framework of use around EVT. Firstly, it takes care of ensuring that the analysis-time distribution (ATD) of the observation measurements upper-bounds the operation-time distribution (OTD). Subsequently, it feeds a sample of ATD, called ATS, to EVT, which can then safely use it to model the tail of the ATD, which in turn is warranted to upper-bound the tail of the OTD. This is better illustrated in Figure 1. The dotted line depicts the empirical complementary cumulative distribution (ECCDF) of the execution-time behaviour of a program, derived from observations of it taken during operation, hence its OTD. We plot this OTD for illustrative purposes only, since its retrospective nature (which observes operation-time events) is too late to feed WCET analysis. The dashed line depicts instead the program's ATD, as resulting from the execution-time conditions in effect during analysis. The solid light line depicts the curve computed by EVT when feeding it with the ATS obtained from the program's ATD. ProMBPTA ensures that ATD upper-bounds OTD, and makes a sound use of EVT to compute high-quality pWCET estimates that

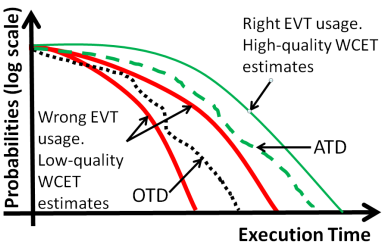

Fig. 1. EVT quality issues.

upper-bound ATD, and hence OTD. An incorrect use of EVT might instead result in low-quality pWCET estimates that may fail to upper-bound OTD while still upper-bounding non-representative ATD.

III. EVT AND WCET: GUMBEL IS SAFE

Before presenting the proMBPTA framework, we discuss the assumptions that characterize the problem domain of WCET analysis and review what they imply for the application of EVT, particularly the way it models the tail of the probability distribution.

A. Introduction on EVT

EVT makes no assumption on the internals of the system (a computing platform in our case) from which the measurements are collected. In our problem domain, EVT has to be understood as a technique to predict the probability distribution of the combined impact of the timing events observed in the provided sample of analysis-time measurements. To contribute to the combined effect that EVT seeks to predict, the base events have to be observed (while their cumulative effect need not). It follows that unobserved events (whose effect on execution time might be arbitrarily large) cannot contribute to the computation [5], [7]: if they can occur and are not captured in the observations, then the predictions are inaccurate and therefore fallacious.

EVT accurately approximates the tail (hence the extreme) of a given probability distribution, taking in input only those observations from the sample that belong to the tail. The Block Maxima (BM) and Peak Over Threshold (PoT) [19] methods are used to that end.

Block Maxima. BM defines a block size (bs) and splits the sample in smaller groups (e.g. a sample of $R = 2,500$ elements and a block size of $bs = 25$ elements yield $nb = \frac{R}{bs} = 100$ blocks of bs observations each). EVT draws the highest value in each block and creates a sample of maxima to which it fits a probability distribution. Three families of continuous probability distributions are used for fitting in conjunction with the BM method: Gumbel, Fréchet and (reversed) Weibull. They are jointly described by the parametric Generalised Extreme Value (GEV) distribution family. The CDF of GEV is defined by the parameters μ, σ and ξ, known as the *location*, *scale* and *shape* respectively [1], [15].

Peak-over-Threshold. PoT defines a threshold (th) value and creates a maxima population drawing from the sample only the observations higher than th. The resulting probability distribution function can be described by the Generalised Pareto Distribution (GPD) family. GPD can be described with either 2 or 3 parameters. In the latter case, μ, σ and ξ, have the same interpretation as for GEV (and ξ is identical): for the same ξ, and similar values for μ and σ, GPD and GEV

yield the same distribution. [15] explains how to determine μ, σ and ξ, and how μ and σ vary for GEV and GPD.

The tail shapes produced by the said distribution families are not equally apt to model WCET behaviour: the (reversed) Weibull family approaches asymptotically an exact (maximum) value, which helps when the WCET is known (but is not our problem domain); the Gumbel family decreases exponentially without converging to a finite upper-bound; the Fréchet family does the same as Gumbel but decreases polynomially, which makes it less tight for our problem domain. The same holds for the GPD model distributions.

B. WCET existence and finiteness

WCET analysis rests on two main assumptions: (i) target programs execute a finite number of instructions; and (ii) each instruction executes in a finite number of cycles. Those assumptions guarantee that a WCET does actually exist and that an upper-bound to it is computable. Parametric static timing analyses have been proposed to soften the finiteness assumptions [20]: while they are useful to reuse results from static analysis, they only produce preliminary formulas that need instantiation before use. A real-time software program is therefore assumed to have a theoretical maximum execution time, the WCET, which needs to be estimated and upper-bounded for safety. For this condition to hold true, the input space of the program itself may require to be known and/or controlled. For example, it would be impossible to derive an upper bound to the execution time of the Factorial function if we did not know or constrain its execution context, in this case the range of admissible input values. We identify two main classes of relevant information on the input ranges that affect the search space, aka *feasible region*, of the WCET:

Finite Execution Conditions (FEC): these are determined by input values that affect loop bounds and/or the depth of recursion, when they are not hard-coded in the program. Excluding parametric methods, which we do not consider in this work, all timing analysis methods, whether static or measurement-based, deterministic or probabilistic alike, need the user to provide some information on FEC. The quality of that information may affect the tightness and trustworthiness of the analysis results. For instance, static timing analyses may require the user to provide *flow facts* to fill in bounding information that could not be automatically derived by, e.g., data flow or range analysis.

Path Traversal Conditions (PTC): when the software program features multiple structural execution paths, identifying the set of feasible and relevant paths is crucial to precision or trustworthiness. This is especially true for measurement-based analyses, which critically depend on the availability of input vectors capable of providing *sufficient* PTC coverage, to warrant cognizant confidence that the paths exercised in the measurement observations include the path leading to the WCET. Static analyses don't need PTC information to achieve full path coverage, but excluding non-relevant or semantically infeasible paths, may improve the quality of their output.

C. Tail distributions and pWCET estimation

FEC&PTC assumptions impact GEV and GPD distribution models, especially for the characterization of the shape parameter ξ. As we noted at the end of section III-A, the reversed Weibull family of distributions should be used when a maximum value exists and it is known. The former hypothesis

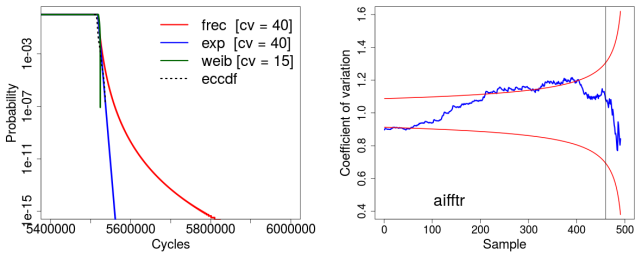

Fig. 2. Effect of misclassified tails.

generally holds for the WCET of real-time software programs; the latter does not. When the WCET is unknown, which is the case in the vast majority of situations in our problem domain, it can be upper-bounded by either a Gumbel or a Fréchet distribution, as the slope of their right tail decreases much more gently than the reversed Weibull one. The descent of the Fréchet distribution is gentler than that of its Gumbel correspondent, and therefore it upper-bounds the latter's right tail. The Fréchet distribution is most appropriate when a maximum value does not exist, which places it outside of the core WCET problem domain. Gumbel, instead, is most appropriate when a maximum value exists but it is unknown.

Methods as the Exponential Test [21] and the Coefficient of Variation (CV) [22] help determine whether the sample fits a Fréchet, Weibull or Gumbel distribution (or their GPD counterparts). Yet, for the argument we have given above (further developed in [23]), the extreme behaviour of the execution-time distribution of a real-time software program can *always* be modelled with a Gumbel distribution: it may not be the tightest one, but it is always – at least – a safe over-approximation of it. Unless we assume that our program of interest has an unbounded WCET, using the Fréchet distribution should be considered inappropriate. If a Fréchet distribution were to best fit a given sample of execution-time measurements, this should happen because either the sample does not contain enough tail values or some of them do not really belong to the tail of the distribution. The remedy would be to increase the sample (to capture more tail values) or use different BM block sizes (to discard less tail values), respectively, as illustrated in [23].

To demonstrate our reasoning, we apply proMBPTA to *aifftr*, a Fast Fourier Transform procedure included in the EEMBC automotive benchmark suite (http://www.eembc.org/benchmark/automotive_sl.php), which we executed 1,000 times on a MBPTA-compliant platform. We first tested exponentiality with the CV test (with 95% confidence): the right part of Figure 2 shows that the exponentiality hypothesis, considered for the 95 highest observations, cannot be rejected, since the blue line (that projects those 95 values) falls within the red lines, meaning that exponentiality cannot be rejected with 95% confidence. In statistical speak, this means that less than 96 maxima observations are sufficient to approximate the Gumbel (exponential) pWCET distribution. Note, however, that passing such test means that data are compatible with the exponentiality hypothesis ($\xi = 0$), but the best fit will likely be non-exponential (although $\xi \approx 0$). As explained before, the decision of using an exponential tail as upper-bound resorts to knowledge about the domain (pWCET estimation), not about the best fit for the data. We obtained fitting pWCET distributions for the three cases plotted in the left part of Figure 2 (using point estimation, although confidence intervals

for the rate of the Exponential distribution could also be considered): (1) Using the 40 highest values yielded a curve that was more a Fréchet than a Gumbel distribution. This is no surprise, in fact, for the probability of obtaining exactly $\xi = 0$ for a sample, as required for a Gumbel distribution, is close to nil; actual sampling typically yields $\xi > 0$ or $\xi < 0$. With 40 values from the sample, we had $\xi = 0.086$, which causes the tail to decrease polynomially (the red line in the plot). (2) Using any other number of values (always less than 96), e.g., 15, yielded $\xi = -0.55$. The resulting pWCET curve was in the domain of the reversed Weibull family (the green line in the plot). As noted earlier, going this way could lead to an unsound, optimistic bound. (3) Picking 40 maxima values and fitting the best Gumbel distribution to them, yielded the tightest and most reliable pWCET curve (the blue line in the plot), which tightly upper-bounds the actual sample (the dashed line) increased to 5,000,000 measurements to prove our point.

Using the best fit distribution from any family in the GEV (or GPD) can lead to arbitrarily pessimistic pWCET estimates. For instance, choosing an exceedance probability of 10^{-15} per run, the pWCET estimate for the test program would be around 5,560,000 cycles for the exponential fit with 40 excesses. However, for 95 values, whose exponentiality cannot be rejected, we obtain a Fréchet distribution and a pWCET of more than 23 million cycles.

Whenever the best fit is a reversed Weibull distribution, we can use a Gumbel equivalent in its stead since, for tail (extreme) behaviour, the Gumbel curve always over-approximates its Weibull correspondent. In other words, the reversed Weibull CCDF may be above the Gumbel one in the first part of the curve, which corresponds to high probabilities, but it is going to be always below it for low probabilities. And only the latter matter in the WCET problem domain.

If the method used to test exponentiality rejects the hypothesis that the tail can be modelled (or upper-bounded) with an exponential tail (e.g. Gumbel or, at least, Weibull), then the sample size needs to be increased until the test passes.

EVT requires the random variable being observed to exhibit a "variable" timing behaviour. Lack of variability prevents EVT from producing good quality models and likely from converging to an exponential distribution. In theory, this can happen regardless of time-randomization when the timing behaviour of the program under analysis has a *degenerate distribution*. While this is extremely rare in practice, at least for real-world programs, it is not necessarily a bad scenario: if representativeness is guaranteed, lack of variability would suggest that the maximum observed execution time could be reasonably regarded as a precise estimator for the WCET.

Conclusions: in general, using EVT without passing this step cannot be considered a valid approach to estimate the WCET of real-time programs: it might be used for other classes of programs characterized, for example, by unbounded execution times (e.g. modeling the execution time distribution of the Factorial function for *any* input value). However, as already discussed, these scenarios are typically not considered as they do not produce directly "usable" WCET estimates.

We therefore conclude that EVT is *adaptable* to all programs that are analyzable by other timing analysis techniques. Moreover, we observed that the tail distributions of this class of programs are properly described by a Gumbel. This contrasts with [13] whose author concludes that the Gumbel family does not always provide a reliable model, resulting in low-quality WCET estimates. In fact, those conclusions were drawn outside of the WCET finiteness and existence assumptions, considering a class of programs that can only be analysed parametrically.

As we noted in Section II, it is the responsibility of the analysis process to guarantee that the statistical and representativeness requirements are met so that EVT can be applied correctly. We now discuss what proMBPTA does to that end.

IV. proMBPTA and EVT Trustworthiness

The accuracy of the model used to represent the timing behaviour of the program of interest is a major factor of influence, not only for the representativeness of observations, but also for how they are collected and fed to EVT. In fact, even when representativeness is assured (which requires collecting a minimum number of runs [**?**], [7], [9]), the way the observations are collected and submitted to analysis critically affects the quality of the EVT results. This is especially evident for the contribution of individual paths to the pWCET distribution of a multi-path program.

In the following, we show how proMBPTA deals with those concerns, and discuss the role that platform-level time randomization and time upper-bounding play in achieving representativeness. In describing an ideal MBPTA approach, we focus first on those programs whose input vectors (or all those regarded as relevant for the WCET) restrict the paths of interest to one. We then consider how the proMBPTA guarantees extend to multi-path programs.

A. Representativeness on Single-Path (Single-Input) Programs

ProMBPTA relies on time-randomization and upper-bounding, applied to jittery hardware resources, to guarantee that measurements capture execution conditions that are no better than those that may occur at system operation [5]. We refer to the factors that produce execution-time variability, as *Sources of Jitter* (SoJ).

Two main steps are needed for assuring representativeness with proMBPTA: (1) singling out the resources with jittery timing behaviour; (2) either randomizing or upper-bounding their response time, to make the analysis-time observations representative of worst-case operation conditions.

Singling out SoJ: SoJ can affect even the simplest of single-path programs. The nature and number of such factors is platform-dependent and have to be determined by an expert. Examples of SoJ include variable-latency FPU operations, cache behaviour, or simple contention effects in a multicore. Each SoJ should be controlled in a manner that guarantees that the timing behaviour observed at analysis time captures the full extent of possible variability and, hence, is representative of system behaviour at operation.

Mastering the impact of all SoJ with standard measurement-based timing analysis techniques is not generally viable. Understanding and explicitly triggering scenarios in which diverse hardware components may have to interact with one another is untenable in practice. Moreover, the increasing complexity of COTS processors causes the cost of any classic timing analysis approach to explode and the quality of their results to decrease. A qualitative analysis of the contribution of hard-to-predict resources is thus the prerequisite to the application of *any* timing analysis technique. For MBPTA,

this effort allows singling out the SoJ, to determine how randomization and upper-bounding can capture their jitter.

A case for Randomization: Representativeness of the diverse SoJ can be attained by injecting randomization in the timing behaviour of selected hardware or software components [5], [17]. Randomization ensures that several of the sources of execution-time variability are *transparently* captured in the analysis results without needed direct user intervention (whether by providing specific input vectors or setting the internal state). Time-randomized resources may include placement and replacement policies in the different cache memories, as well as arbitration policies in shared resources such as interconnection networks (e.g. buses, trees) and memory controllers.

A proMBPTA-compliant platform – which addresses the SoJ concerns as described above – meets the EVT statistical requirements for single-path programs. Each, conveniently collected (see [5], [17]), execution-time measurement corresponds to an independent and identically distributed (i.i.d.) observation of a random variable. Hence, if all requirements are met by construction, EVT can be applied to that problem space safely. Note that EVT is not applied to the real distribution (the full universe of values), but on a sample of it. Hence, although with proMBPTA the required properties hold on the full population, they need to be empirically confirmed to hold for the specific sample, with appropriate i.i.d. tests [2], [24].

Once the exponential test and i.i.d. tests are passed, and the representativeness constraint holds, the distribution yielded by the sample can be regarded as a valid pWCET distribution.

B. Limits of randomization

It has been suggested that randomization alone does not suffice to enable the use of MBPTA [13]. As already observed, while randomization might not cure degenerate distributions, this is not a problem as long as the collected observations are truly representative. We insist, however, that an inattentive use of randomization does not guarantee representativeness.

Applying time randomization to individual processor components (e.g. the cache) makes their jitter MBPTA-compliant. However, doing that on a single resource does not cover the jitter of other resources: for example, [13] notes that randomizing the cache does not cover the jitter caused by FPU or by multiple program paths. Instead, all SoJ have to be studied individually and a solution has to be proposed to address each of them.

C. Probability Distribution of Multi-Path Programs

When the program of interest has multiple execution paths, the concept of representativeness extends to *whether* and *how frequently* each path is traversed by the input data provided and, thus, captured in the analysis.

Representativeness of execution paths: Assuring sufficient path coverage is a widely acknowledged requirement of all timing analysis approaches based on measurements [25], [26]. In contrast with static timing analyses, the results computed by measurement-based methods hold only for the execution paths observed at analysis. Measurement-based analyses typically assume the availability of PTC information to guide the identification of a subset of structurally feasible paths to be included in the analysis. Path identification and generation of the respective input vectors remain a non-trivial problem for the end users. When PTC information is not available (or is considered unreliable), it is still possible to build a synthetic upper-bounding path [27] or use the measurements from several paths to artificially derive measurements for all unobserved paths [26].

Modelling the distributions of several paths: The critical question is whether the timing behaviour of a multi-path program can still be modeled as the outcome of a single random variable, or else each path in the program needs to be considered separately. In the former case, indistinctly feeding all measurements to EVT on account that all paths concur to a unique distribution postulates that the distribution occurred at analysis does match what will happen at operation. While there has been some works on timing analysis based on *a priori* knowledge of path frequency [28], assuming that the analysis-time observations always match the eventual distribution at operation is very hazardous. Even if we assume that users can provide input vectors capable of triggering all execution paths of interest for WCET estimation, we cannot expect them to reason about their probability of occurrence at operation.

Instead, we observe that each single path potentially leads to a different execution time distribution. This is especially evident in programs with significant input-data dependence (and operation modes), where observations may fit significantly different distributions depending on the provided input vectors. Putting all measurements into a single bucket and applying EVT simple-mindedly on them is not advisable, as the collected measurements, while possibly independent, do not necessarily have identical distribution. Even seeking fair path frequency at analysis, for example by enforcing *uniformly distributed inputs* [29], does not solve the problem that the path distribution at operation is generally unknown and thus cannot be modeled.

Hence, one option is to study each execution path in isolation and derive an envelope distribution that safely accounts for all considered paths. We refer to this solution as *multiple-bucket* application of EVT, as opposed to its single-bucket alternative. We do not consider how these paths have been selected, whether by the user or by other means [26], [27]. In this scenario, the whole MBPTA process is applied on a per-path basis to obtain individual CCDFs. Then a *Max-Envelope* can be computed taking the maximum value for each exceedance threshold across the pWCET distributions of all the considered paths. If the cost of considering each path separately is high, the user can apply path upper-bounding [27] and then analyze any of the modified paths, which is an upper-bound of all possible traversed paths.

Considering measurements from different paths in the same bucket can potentially lead to unreliable results that can equally be unsafe or more pessimistic than those obtained with the Max-Envelope method, which is safe by construction. We illustrate this on a real case study from the railway domain. We experimented with the signalling subsystem, responsible for protecting the train motion in a simplified European Train Control System (sETCS), running on a LEON3-based FPGA implementation, with 16KB randomized Instruction and Data caches. We collected observations for 10 input vectors, which corresponded to 10 distinct paths in the programs. To demonstrate the inappropriateness of the single-bucket approach, we compared the results of applying MBPTA on each path in isolation against the results from pairwise combination of different paths. The left side of Figure 3 shows that combining samples from paths 0 (yellow) and 3 (blue) in a single bucket

causes the computed distribution (red) to fall even behind the distribution of each individual path considered in isolation. The right side of Figure 3 shows the combined distribution (red) obtained by merging the samples from paths 7 (yellow) and 9 (blue). In this case, the single-bucket approach causes the final combined distribution to fall largely beyond the Max-Envelope values (in this case corresponding to the yellow line).

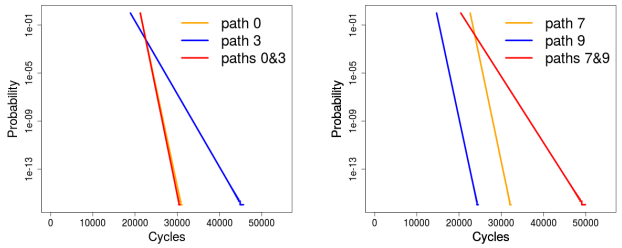

Fig. 3. Unsafe and pessimistic results when combining paths.

V. Related Work

After a seminal paper [30], a lot of effort has been devoted to understand the challenges of applying statistical approaches to WCET analysis [31]–[33].

Several efforts have been directed to studying the statistical requirements and support needed to effectively use EVT. A primal claim in this sense was that EVT requires observations to be i.i.d. [32], [33]. Randomization has been proposed as a means to facilitate the use of EVT [5], [7]–[9]. Randomization guarantees that the i.i.d. requirement is satisfied by construction [17] and allows probabilistic reasoning on the representativeness of input data and the quality of results [9].

The i.i.d. requirement has been softened [6], [15], [34] showing that EVT can be applied on data exhibiting stationary or weak dependence. This observation led to the definition of well-structured approaches to apply statistical tests on the sample data and to assess the goodness of model fitting [34]. This has paved the way for the application of EVT tools to programs running on deterministic (i.e., non-randomized) hardware [6], [35]. However, this use fails to account for the role and importance of randomization in sustaining the representativeness of inputs.

Finally, the application of EVT to model multiple paths, relating to path concerns in measurement-based analyses, has been studied in [26], [27] where methods have been proposed to increase the eventual path coverage.

VI. Conclusions

In this paper, we argued that the main misconceptions on the application of EVT to the WCET problem can be defeated by considering the peculiarities of execution time as the event EVT is expected to model. With a view to the application of EVT to industrial-quality programs, we demarcate proMBPTA, a well-defined, rigorous analysis process that guarantees a WCET-centric application of EVT with guarantees on trustworthiness of its results and wide adaptability to software programs.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (MINECO) under grant TIN2015-65316-P and the HiPEAC Network of Excellence. Jaume Abella has been partially supported by the MINECO under Ramon y Cajal grant RYC-2013-14717. Authors also thank George Lima for his feedback on this manuscript.

References

[1] S. Kotz *et al.*, *Extreme value distributions: theory and applications*. World Scientific, 2000.
[2] L. Cucu-Grosjean et al, "Measurement-based probabilistic timing analysis for multi-path programs," in *ECRTS*, 2012.
[3] F.J. Cazorla et al, "PROARTIS: probabilistically analysable real-time systems," *ACM Transactions on Embedded Computing Systems*, 2012.
[4] J. Abella et al, "WCET analysis methods: Pitfalls and challenges on their trustworthiness," in *SIES*. IEEE, 2015, pp. 1–10.
[5] F. Cazorla *et al.*, "Upper-bounding program execution time with extreme value theory," in *WCET Workshop*, 2013.
[6] L. Santinelli *et al.*, "On the sustainability of the extreme value theory for WCET estimation," in *WCET Workshop*, 2014.
[7] J. Abella *et al.*, "Heart of Gold: Making the improbable happen to extend coverage in probabilistic timing analysis," in *ECRTS*, 2014.
[8] E. Mezzetti *et al.*, "Randomized caches can be pretty useful to hard real-time systems," *LITES*, vol. 2, no. 1, pp. 01:1–01:10, 2015.
[9] S. Milutinovic et al., "Modelling probabilistic cache representativeness in the presence of arbitrary access patterns," in *ISORC*, 2016.
[10] F. Wartel *et al.*, "Measurement-based probabilistic timing analysis: Lessons from an integrated-modular avionics case study," in *SIES*, 2013.
[11] M. Fernandez *et al.*, "Probabilistic timing analysis on time-randomized platforms for the space domain," in *DATE*, 2017.
[12] F. Cros *et al.*, "Dynamic software randomisation: Lessons learned from an aerospace case study," in *DATE*, 2017.
[13] G. Lima *et al.*, "Extreme value theory for estimating task execution time bounds: A careful look," in *ECRTS*, 2016.
[14] J. Reineke, "Randomized caches considered harmful in hard real-time systems," *LITES*, vol. 1, no. 1, pp. 03:1–03:13, 2014.
[15] S. Coles, *An introduction to statistical modeling of extreme values*, ser. Springer Series in Statistics. Springer-Verlag, 2001.
[16] S. Law *et al.*, "Achieving appropriate test coverage for reliable measurement-based timing analysis," in *ECRTS*, 2016.
[17] L. Kosmidis *et al.*, "Fitting processor architectures for measurement-based probabilistic timing analysis," *Microprocessors and Microsystems*, vol. 47, Part B, pp. 287 – 302, 2016.
[18] M. Slijepcevic *et al.*, "DTM: Degraded test mode for fault-aware probabilistic timing analysis," in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2013.
[19] W. Feller, *An Introduction to Probability Theory and Its Applications*. John Willer and Sons, 1996.
[20] S. Bygde *et al.*, "An efficient algorithm for parametric WCET calculation," *Journal of Systems Architecture*, vol. 57,6, pp. 614 – 624, 2011.
[21] M. Garrido *et al.*, "The ET test, a goodness-of-fit test for the distribution tail," in *Methodology, Practice and Inference, second international conference on mathematical methods in reliability*, 2000, pp. 427–430.
[22] J. Del Castillo *et al.*, "Methods to distinguish between polynomial and exponential tails," *Scandinavian Journal of Statistics*, vol. 41, no. 2, pp. 382–393, 2014.
[23] J. Abella et al., "Measurement-based worst-case execution time estimation using the coefficient of variation," *ACM Transactions on Design Automation of Electronic Systems*, to appear 2017.
[24] J. Abella *et al.*, "Extreme value theory in computer sciences: The case of embedded safety-critical systems," in *6th International Conference on Risk Analysis (ICRA)*, 2015.
[25] R. Wilhelm *et al.*, "The worst-case execution time problem: overview of methods and survey of tools," *Trans. on Embedded Computing Systems*, vol. 7, no. 3, pp. 1–53, 2008.
[26] M. Ziccardi *et al.*, "EPC: Extended path coverage for measurement-based probabilistic timing analysis," in *RTSS*, 2015.
[27] L. Kosmidis *et al.*, "PUB: Path upper-bounding for measurement-based probabilistic timing analysis," in *ECRTS*, 2014.
[28] L. David *et al.*, "Static determination of probabilistic execution times," in *ECRTS*, June 2004, pp. 223–230.
[29] Y. Lu *et al.*, "A statistical response-time analysis of real-time embedded systems," in *RTSS*, 2012.
[30] S. Edgar *et al.*, "Statistical analysis of WCET for scheduling," in *the 22nd IEEE Real-Time Systems Symposium (RTSS01)*, 2001.
[31] G. Bernat *et al.*, "WCET analysis of probabilistic hard real-time system," in *RTSS*, 2002.
[32] J. Hansen *et al.*, "Statistical-based wcet estimation and validation," in *WCET Workshop*, 2009.
[33] D. Griffin *et al.*, "Realism in Statistical Analysis of Worst Case Execution Times," in *WCET Workshop*, 2010.
[34] F. Guet *et al.*, "On the reliability of probabilistic worst-case execution time estimates," in *ERTS*^{2}, 2016.
[35] K. Berezovskyi *et al.*, "Measurement-based probabilistic timing analysis for graphics processor units," in *ARCS*, 2016.